Аппаратная поддержка виртуализации вычислительных ресурсов

Алексей Кожин

Введение

- **Виртуализировать** компьютер:
  - Представить реальный компьютер в виде нескольких виртуальных компьютеров (*виртуальных машин*) или виртуального компьютера с другой архитектурой
  - Представить несколько физических компьютеров как один отдельный виртуальный вычислительный модуль (серверный кластер, grid computing)
- Первые виртуальные машины были разработаны еще в 1960-х IBM вместе с развитием машины System 360/67, а аппаратная поддержка виртуализации появилась уже в архитектуре следующего семейства IBM System 370. Каждому пользователю предоставлялся точный виртуальный образ аппаратных ресурсов физической машины, и обеспечивалась работа в режиме разделения времени.
- Для машин IBM System 370 был разработан монитор виртуальных машин, который осуществлял перехват и эмуляцию потенциально опасных команд с помощью аппаратных средств расширенной архитектуры.

Виртуальная машина

- **Виртуальная машина (Virtual Machine, VM)** — полностью защищенная и изолированная копия ресурсов физической машины, в которой приложения и операционные системы ведут себя точно так же как, и на реальных аппаратных средствах.
- На одной реальном компьютере может быть запущено множество виртуальных машин, каждая со своей операционной системой.
- Любая гостевая операционная система рассматривает базовую аппаратуру как принадлежащую ей и не знает о существовании других ОС. На самом деле эту иллюзию создает у них гипервизор.

Гипервизор

Монитор Виртуальных Машин (Гипервизор, Virtual Machine Monitor, VMM) берет на себя управление гостевыми системами. Его можно рассматривать как абстракцию между аппаратной платформой и виртуальными машинами.

В некоторых случаях гипервизор является операционной системой; в этом случае он называется базовой операционной системой.

Гипервизор обеспечивает распределение ресурсов host-системы между гостевыми ОС: виртуализацию процессора, памяти и устройств ввода/вывода, а также изоляцию программ, перехват и эмуляцию потенциально опасных команд.

Его функциональность зависит от типа виртуализации.

Уровневая модель виртуализации

Типы виртуализации

- Эмуляция оборудования
- Полная виртуализация
- Паравиртуализация
- Виртуализация уровня операционной системы

Эмуляция оборудования

- Гостевая ОС запускается на виртуальной машине, которая перехватывает каждую команду и моделирует ее на реальном аппаратном обеспечении.
- Нет привязки к архитектуре host-машины (можно запускать неизмененные операционные системы, предназначенные для машин с другой архитектурой).
- Очень низкая скорость работы.

Полная виртуализация

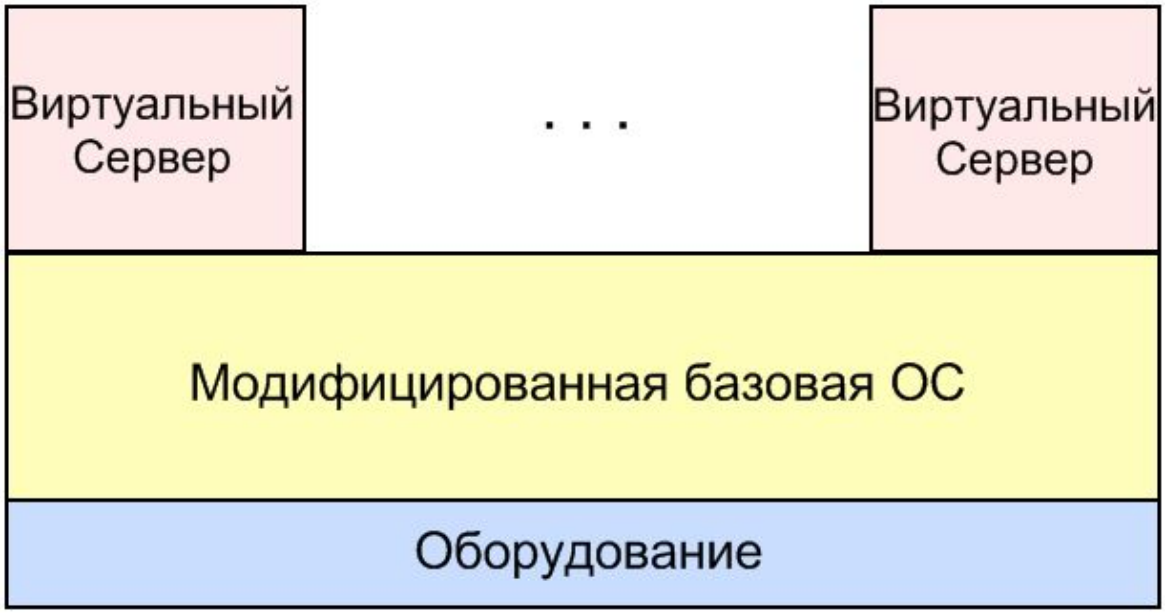
- Гипервизор обеспечивает связь между реальным оборудованием и гостевыми ОС; он управляет виртуальными машинами и разделяет между ними аппаратные ресурсы.
- Монитор виртуальных машин перехватывает только некоторые опасные команды, все остальные напрямую выполняются на аппаратном обеспечении.
- Скорость работы выше, чем при эмуляции, но из-за потерь в гипервизоре производительность меньше, чем при отключенной виртуализации.
- Гостевая операционная система должна поддерживать аппаратные средства host-машины.

Паравиртуализация

- Используется гипервизор для связи между аппаратурой и гостевыми ОС.
- Ядро ОС необходимо модифицировать: вместо опасных инструкций оно должно выполнять гипервызовы (hypercalls) на обработку гипервизором. Библиотеки и приложения уровня пользователя не изменяются.
- Гипервизор предоставляет гостевой ОС специальный API, с которым она и взаимодействует, вместо того чтобы обращаться напрямую к таким ресурсам, как таблица страниц памяти.
- Производительность выше, чем при полной виртуализации.
- Гостевая операционная система должна поддерживать аппаратные средства host-машины.

Виртуализация уровня операционной системы

- Поддерживается только одна базовая ОС с модифицированным ядром, в которую встроен гипервизор. Она сама разделяет аппаратные ресурсы и управляет виртуальными серверами.
- Все виртуальные серверы работают в одной операционной системе, но изолированы друг от друга.
- Производительность близка к производительности невиртуализованной системы.



Предпосылки внедрения виртуализации

- Острый интерес к технологиям виртуализации в 2000-х.
- Главной предпосылкой стал постоянный рост инфраструктуры ИТ, как следствие – увеличение затрат на электроэнергию и обслуживание. При этом средний коэффициент загруженности серверов составлял всего 25%.
- Другой побудительный мотив состоит в том, чтобы сделать инфраструктуру максимально управляемой. Экономия достигается за счет более эффективного использования оборудования, и в меньшей степени — благодаря сокращению персонала, занимаемых площадей и оптимизации работы ПО.

Использование виртуализации в настоящее время

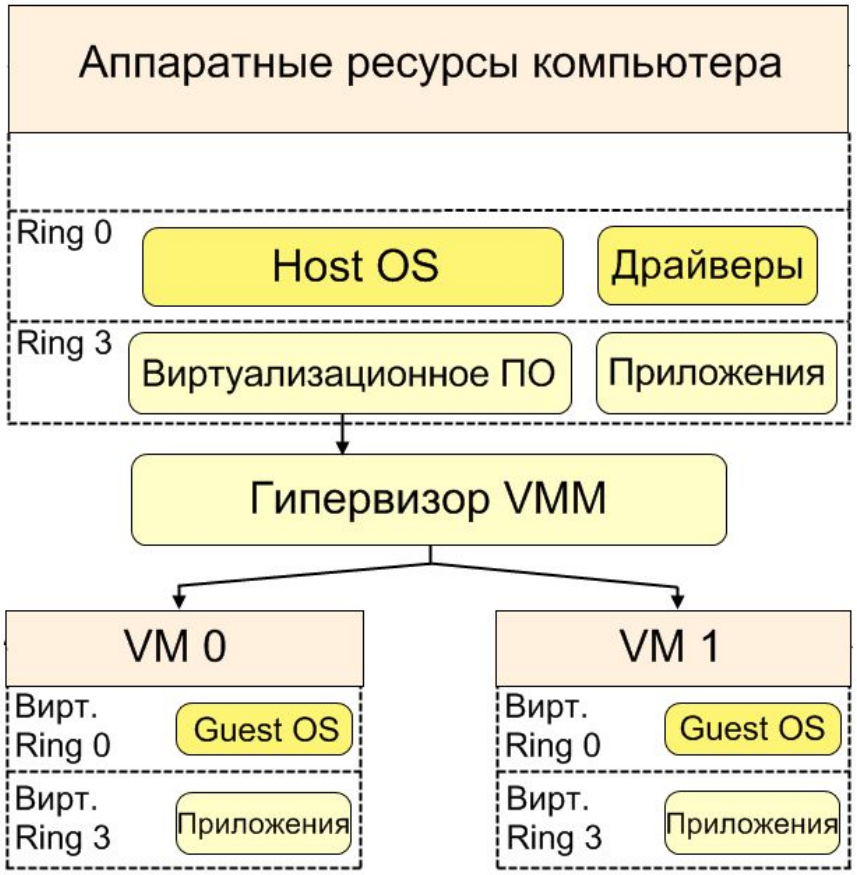
- **Серверная консолидация** - виртуализация множества недостаточно использованных систем на отдельном сервере. Позволяет увеличить средний коэффициент загруженности серверов на 20% и сэкономить на мощности, месте, охлаждении и администрировании из-за наличия меньшего количества серверов.
- Технология EMC VMotion в VMware VirtualCenter дает возможность осуществить **живую миграцию (live migration)** – динамически перенести операционные систем вместе со всеми выполняющимися приложениями между различными серверами без их остановки, обеспечивая непрерывную доступность сервисов
  - позволяет сбалансировать нагрузку на доступном оборудовании либо предотвратить остановку выполнения задач при его отказе
  - полностью сохраняется состояние оперативной памяти как для ядра ОС, так и для выполняющихся приложений
  - сохраняются все установленные соединения и не требуется повторное подключение активных пользователей.

Аппаратная поддержка виртуализации

- Технологии:
  - **Intel VT**
  - **AMD Pacifica**
- Используют идеи и решения, отработанные на виртуальных системах IBM VM (начиная с IBM System 370 – первой машины, обладающей аппаратной поддержкой VM)
- Поддержка гипервизоров при полной виртуализации и паравиртуализации

Intel Virtualization Technology for x86 (VT-x)

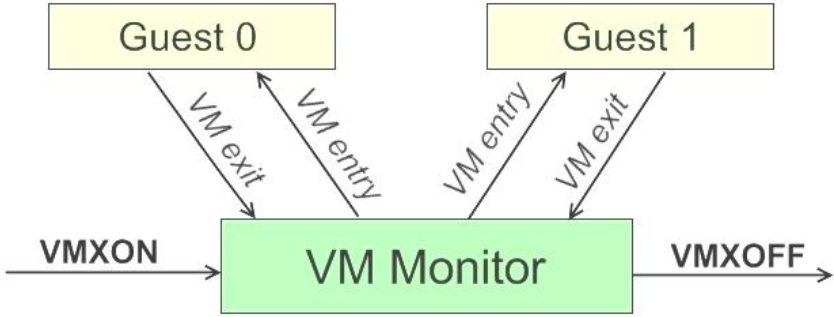
- Гипервизор (VMM) запускается как приложение базовой ОС (VMWare Workstation и прочие), все виртуальные машины (VM) с гостевыми ОС управляются VMM.
- Специальный режим работы процессора - режим выполнения виртуальной машины **Virtual Machine eXecution Mode (VMX).**
- В режиме VMX два типа выполнения программного кода: **VMX-root** и **VMX-non-root**.
- **VMX-root** выполняется на процессоре практически так же, как и вне режима VMX, за исключением того, что в данном режиме предусмотрены дополнительные наборы инструкций процессора, а также некоторые ограничения на значения, загружаемые в управляющие регистры. Гипервизор всегда выполняет свой код как VMX-root.



- **VMX-non-root** предусмотрен для гостевой операционной системы. Взаимодействие гостевой операционной системы с процессором происходит посредством гипервизора. Вместо некоторых инструкций - выполнение выхода из виртуальной машины и передача управления гипервизору. VMX-режим для кода VMX-non-root не требует каких-либо модификаций гостевой ОС.

Intel VT: Гипервизор

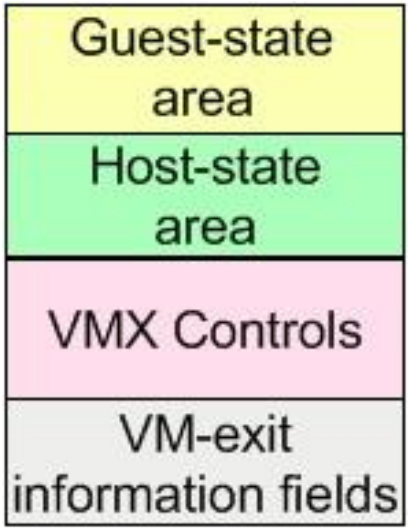
- Запуск / выход из режима VMX (и гипервизора) командами **VMXON** / **VMXOFF**
- В режиме VMX два перехода:
  - *VM entry* – переход VMX-root => VMX-non-root
  - *VM exit* – переход VMX-non-root => VMX-root



- Запуск / продолжение выполнения выбранной виртуальной машины командами **VMLAUNCH** / **VMRESUME** в режиме VMX-root
- Виртуальная машина (а также переходы *VM entry* и *VM exit*) описывается структурой **VMCS**
- Выход из исполнения и передача управления VMM:
  - Безусловный выход (осуществляется независимо от настроек VMCS) при вызове:
    - некоторые инструкции режима VMX-non-root (CPUID, GETSEC, INVD, XSETBV)
    - команды выхода из виртуальной машины **VMCALL**
    - всех остальных инструкций VMX
  - Выход по условию (настраивается в VMCS) при:
    - доступе к некоторым процессорным регистрам (CR0, CR3, CR4, CR8)
    - доступе к пространствам ввода/вывода, APIC
    - перехвате исключений и прерываний
    - при перехвате многих инструкций VMX-non-root

Intel VT: VMCS

- Структура **VMCS** (**Virtual Machine Control Structure**) – небольшой участок оперативной памяти (несколько KB), где хранятся:
  - данные, необходимые для запуска гостевой операционной системы (*guest-state area*)
  - данные, требуемые для безопасного выхода из режима работы гостевой ОС (*host-state area*)
  - поле *VMX Controls*, содержащее:
    - условия *VM exit*
    - маски для регистров CR0 и CR4, позволяющие гостевой ОС осуществлять доступ только к разрешенным битам этих регистров
    - Virtual-Processor identifier (VPID)
    - Extended-Page-Table Pointer (EPTP)
    - некоторые другие настройки
  - информация о причине последнего VM exit (*VM-exit information fields*)



Каждому логическому процессору соответствует своя структура VMCS и указатель на нее

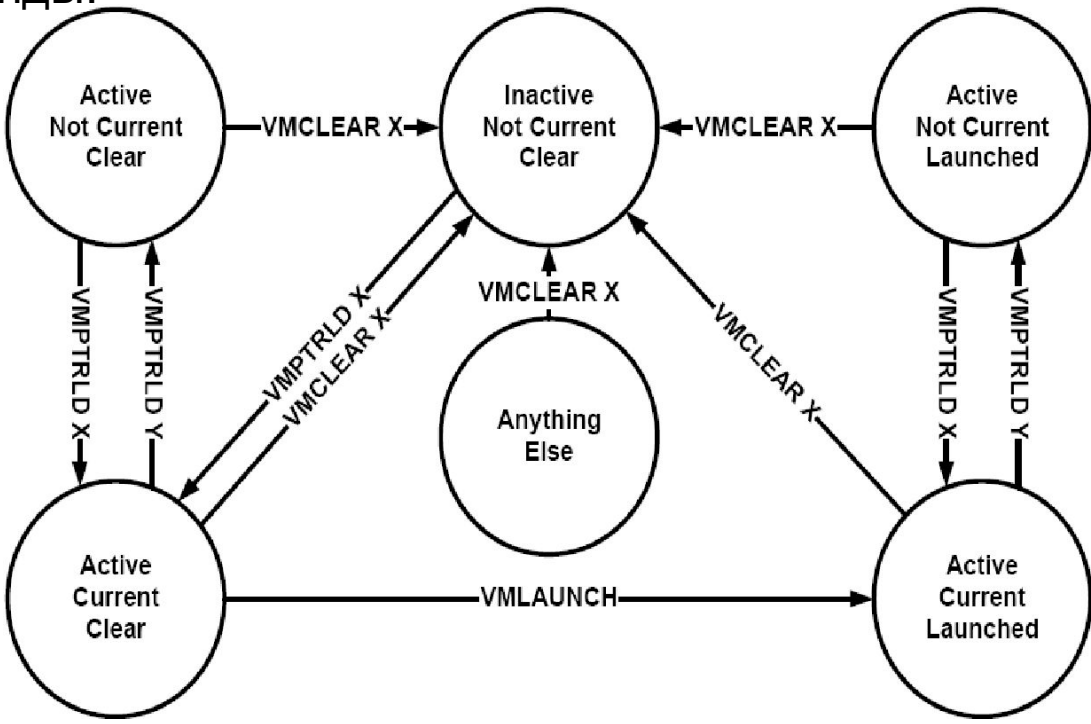
Intel VT: VMCS States

VMCS каждой виртуальной машины может находится в нескольких состояниях:

- *Active/Inactive* – она существует (возможно пустая) либо нет
- *Current/Not Current* – из нее загружается текущая VM либо нет (не более одной)
- *Launched/Clear* – в нее были записаны данные VM либо она еще пустая

Введены следующие VMX-root команды:

- **VMWRITE / VMREAD** – запись/чтение полей текущей структуры VMCS
- **VMPTRLD** – выбор текущей VM (указатель на VMCS)
- **VMPTRST** – сохраняет значение указателя на текущую VMCS в специальной области памяти
- **VMLAUNCH** – запуск VM из текущей VMCS
- **VMCLEAR** – инициализация пустой VMCS либо перевод выбранной VM в остановленное состояние



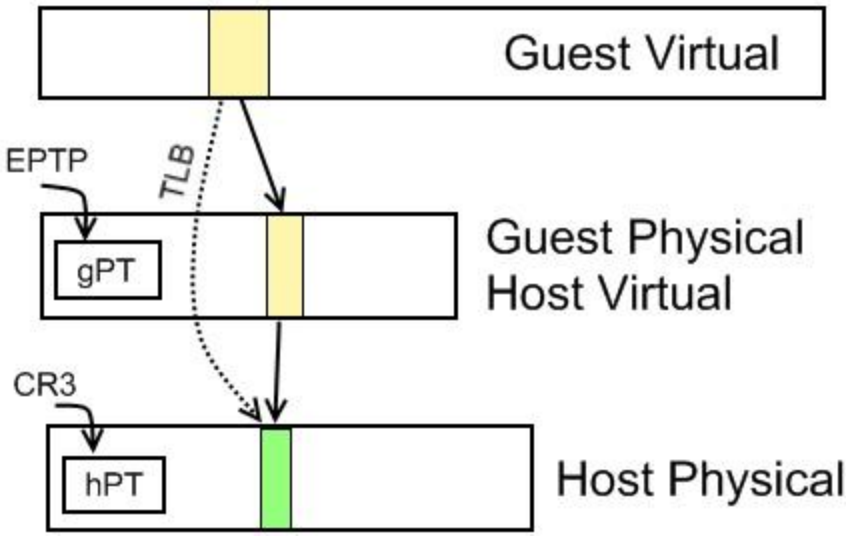
Состояния структуры X (качестве параметров – указатель на VMCS)

Intel VT: VM entry & VM exit

- При *VM entry*:
  - проверка возможности загрузки VM
  - в *host-state area* записывается текущее состояние гипервизора
  - загружается последнее состояние гостевой ОС из *guest-state area*
  - если была вызвана инструкция VMLAUNCH – изменение состояния VMCS
- При *VM exit* – перехвате событий, описанных в поле *VMX Controls*, либо при безусловном выходе:
  - записывается причина выхода из исполнения виртуальной машины в поле *VM-exit information fields*
  - записывается текущее состояние VM в *guest-state area*
  - загружается последнее состояние гипервизора из *host-state area*
- Есть возможность управления счетчиком тактов и показанием системного таймера
- В случае ошибки при переходах - *shutdown*

Intel VT: Address Translations

- **Virtual-Processor identifier (VPID)** – 16-битовый идентификатор виртуального процессора (0 для VMX-root и вне VMX); используется для того, чтобы не выполнять *flush TLB* при каждом переключении VM. При кэшировании в TLB выполняет роль тэгов; загружается из VMCS.
- **Extended Page Table mechanism (EPT)** – механизм виртуализации физической памяти. Каждая виртуальная машина имеет свою собственную виртуальную память. Физическая память гостевой ОС является областью виртуальной памяти базовой ОС. **EPTP** – указатель на таблицу страниц VM, загружается из VMCS. В TLB с соответствующим тэгом VPID помещается преобразование виртуального адреса гостевой ОС в физический базовой ОС.



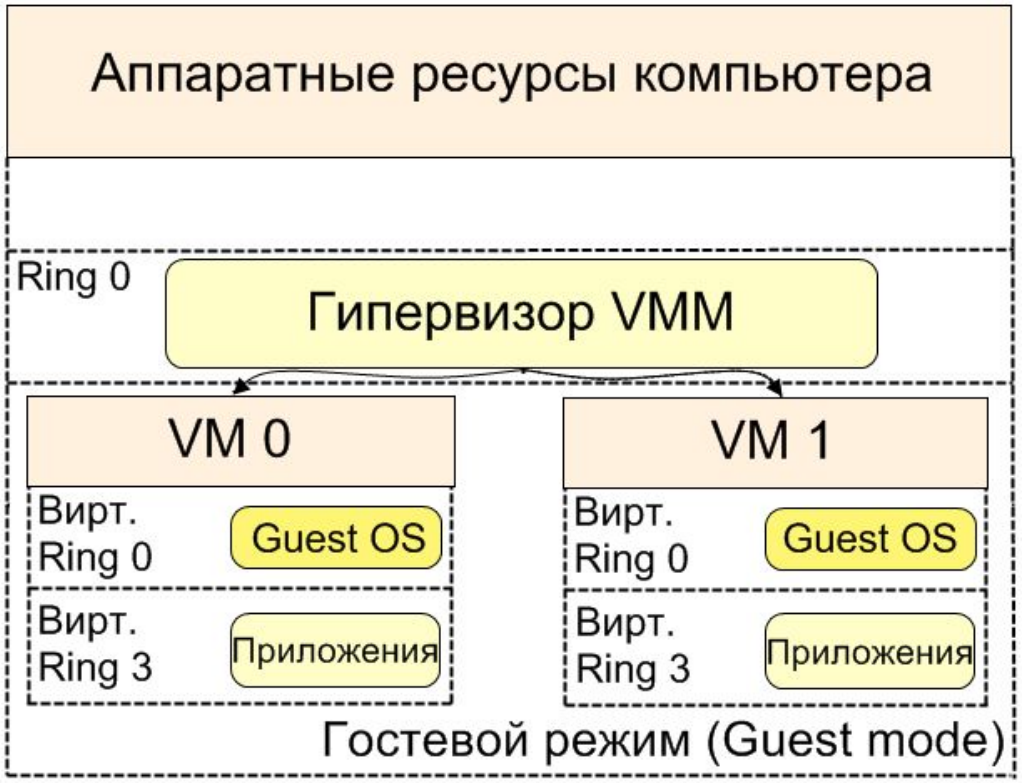
Intel: резюме

- Добавлены 10 новых инструкций: **VMXON**, **VMXOFF**, **VMLAUNCH**, **VMRESUME**, **VMCALL**, **VMWRITE**, **VMREAD**, **VMPTRLD**, **VMPTRST**, **VMCLEAR**
- Гипервизор является приложением базовой ОС
- Добавлена работа со структурами данных **VMCS**, описывающими переходы между гипервизором и виртуальными машинами; эти структуры хранятся в оперативной памяти
- Добавлен механизм **EPT**, поддерживающий виртуализацию физической памяти, и **VPID** – идентификатор виртуального процессора, позволяющий не выполнять очистку TLB при смене виртуальной машины

AMD Pacific

- Расширение **Security and Virtual Machine (SVM)**
- Гипервизор является системным кодом, запускаемым из нулевого уровня, и выполняет роль ядра некоторой основной операционной системы и только он работает с физическим оборудованием.

- В AMD Pacific нет никакого различия между host ОС и гостевыми ОС, как в технологии Intel VT. Все операционные системы считаются гостевыми и для каждой из них гипервизор создает свою виртуальную машину.



AMD Pacific: Запуск VM

- Запуск командой **VMRUN** [указатель на **VMCB** вызываемой VM]
- **VMCB (Virtual Machine Control Block)** – область оперативной памяти (несколько KB), где хранятся:
  - список событий и инструкций для перехвата гипервизором
  - управляющие регистры, определяющие доступ к физическим ресурсам процессора (виртуальная память, пространство I/O и др.)
  - *guest processor state* (последнее состояние запускаемого виртуального процессора)
  - причина последнего выхода из VM (поле *EXITCODE)*
- В отличие от VMCS *в VMCB не хранится состояние host процессора*; оно частично либо полностью записывается во внутреннюю память процессора (hidden on-chip memory), а также может частично храниться в специально отведенной области физической памяти
- После вызова VMRUN:
  - Сохранение *host state*
  - Проверка и загрузка *guest state*
  - Загрузка управляющих регистров
  - В случае обнаружения ошибки – событие **#VMEXIT** и возврат к host

AMD Pacific: Выход из VM

- Выход осуществляется при перехвате определенных событий (заданных в VMCB) либо при вызове инструкции **VMCALL**
- При выходе (событие ***#VMEXIT***):
  - Отключение перехвата прерываний (очистка *GIF – Global Interrupt Flag*)
  - Запись состояния виртуального процессора в *guest state* VMCB
  - Запись причины выхода в поле *EXITCODE*
  - Загрузка *состояния host* процессора
  - Очистка некоторых регистров процессора
- Сохранение и загрузка дополнительных параметров в структуре VMCB с помощью команд **VMSAVE** / **VMLOAD**
- Для ускорения работы также можно использовать инструкции **STGI** / **CLGI** – включение / отключение перехвата прерываний

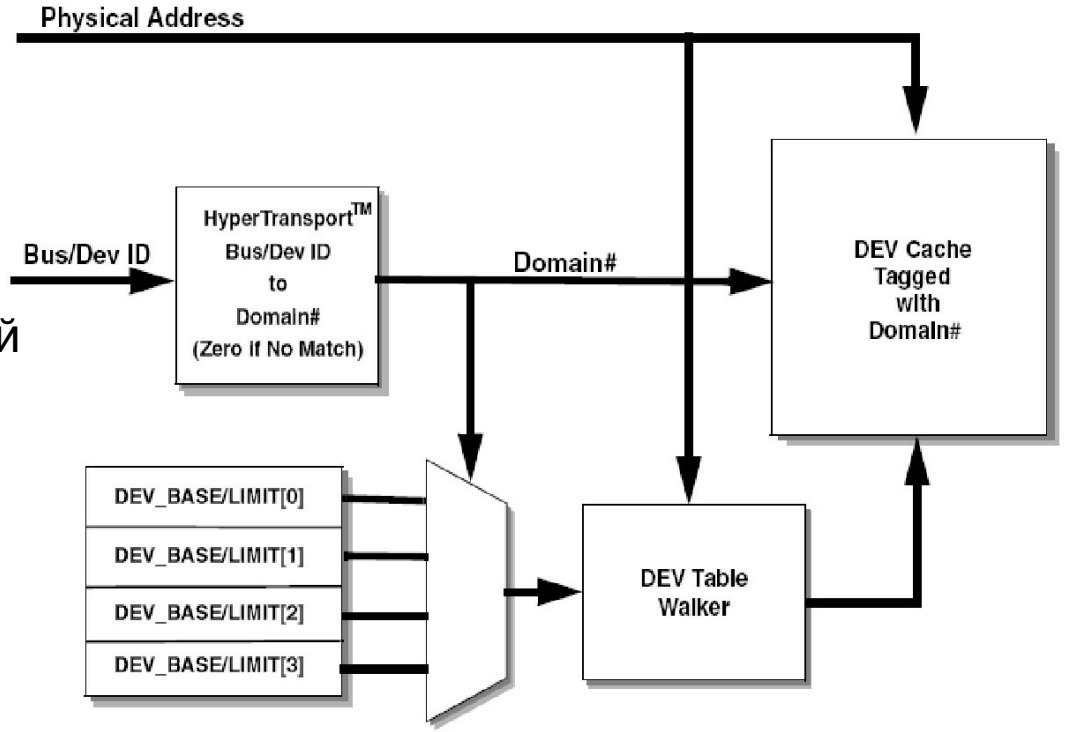
AMD Pacific: Управление доступом в память

- Механизм **Protection Domains** управляет доступом по DMA всех для всех устройств, подключенных к шине PCI или HyperTransport, реализован в North Bridge (NB, вынесенный). Выделяются несколько доменов (по-умолчанию 4) с различными правами доступа к страницам памяти, для каждого домена есть своя область физической памяти ***DEV (Device Exclusion Vector).***
- *DEV – непрерывный* массив битов физической памяти, каждый бит определяет доступ к соответствующей 4KB странице физической памяти
- Для улучшения производительности *DEV* может кэшироваться как и вся остальная память; кэширование этой области памяти можно отключить в регистре *DEV_CR*
- У каждой устройства – свой ***Device ID***, соответствия *Device ID* => домен хранятся в NB

AMD Pacific: Управление доступом в память

- Доступ устройств в память осуществляется следующем образом:
  - Прием запроса в host bridge
  - Получение *Device ID* и выбор соответствующего *DEV*
  - Индексирование *DEV* частью адреса (отбрасываются младшие 12 бит => 4KB)
  - Проверка выбранного бита: если 1, то запрещен доступ на чтение / запись в эту страницу
  - Возврат сообщения о об ошибке, если доступ запрещен

- При доступе в память через host bridge всегда происходит проверка в *DEV*, при запросе из другого NB (в многопроцессорной системе) права на доступ не проверяются

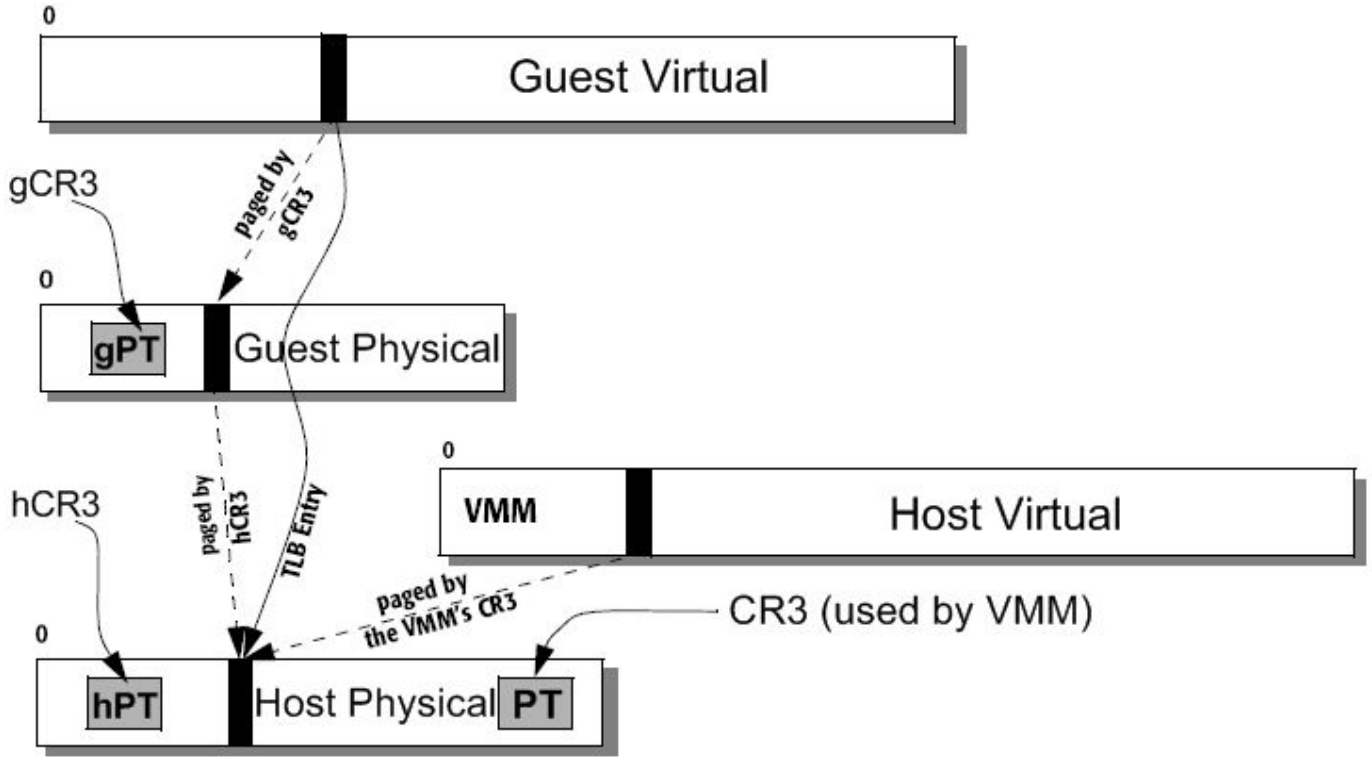


AMD Pacific: Управление доступом в память

- I/O Space Access: можно запретить доступ устройств к пространству I/O с помощью установки соответствующего бита в конфигурационных регистрах *DEV*; передачи между North Bridges не проверяются
- Config Space Access: доступ заблокирован для всех запросов, кроме запросов от Home-процессора

AMD Pacific: Nested Paging

- **Nested Paging** – трансляция виртуальных адресов в физические без вмешательства VMM (если при этом не происходит *#VMEXIT*)
- Два уровня трансляции:
  - **Guest Page Table (gPT)** – преобразование *guest virtual address => guest physical address*; находится в физической памяти VM, имеет свой gCR3
  - **Host Page Table (hPT)** – преобразование *host virtual address => host physical address*; находится в физической памяти процессора, имеет свой hCR3
- После первого уровня трансляции с помощью **gPT** *guest physical address* транслируется дальше как *host virtual address* с **hPT**



AMD Pacific: Nested Paging

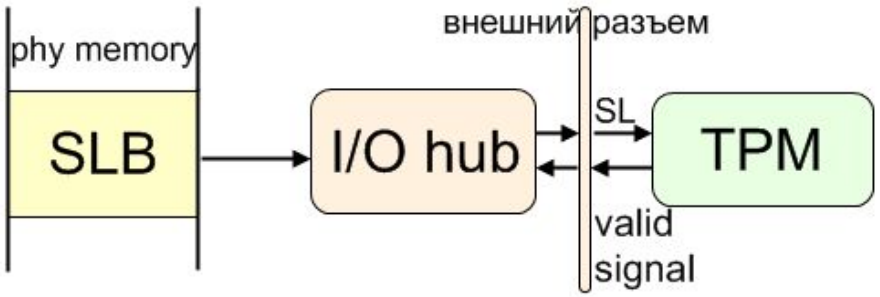
- Для ускорения переключения между виртуальными машинами и гипервизором каждую строку TLB дополняют тэгами – идентификаторами адресного пространства (*address space identifier, ASID*). Это позволяет избежать сброса буфера при каждом входе и выходе из виртуальной машины.
- Результат трансляции записывается в TLB вместе с идентификатором текущей виртуальной машины или VMM.
- Для ускорения работы можно очистить записи TLB только для текущей VM или VMM с помощью инструкции **INVLPGA.**

AMD Pacific: Защищенный режим

- **Защищенный загрузчик** (**secure loader, *SL***) – компонент ПО, который после выполнения проверки инициализирует механизм SVM со всеми соответствующими структурами данных, а также запускает доверенное ПО (гипервизор VMM)
- Выполняется проверка, что загрузчик имеет цифровую подпись доверенного источника; сам загрузчик в свою очередь может проверять подпись гипервизора, гипервизор – подпись ОС и так до любой степени контроля
- Весь код *SL* хранится в области физической памяти процессора, называемой **SLB** (**secure loader block**) и имеющей размер 64 KB
- Проверка загрузчика выполняется во внешнем устройстве **trusted platform module** (**TPM**), который подключается через внешний разъем к I/O hub'y; это устройство вычисляет значение хэш-функции для входных данных, сравнивает его со своим эталоном и возвращает результат сравнения в процессор
- Запуск защищенного режима осуществляется с помощью инструкции **SKINIT**; в качестве ее параметра – указатель на SLB (код загрузчика уже должен быть помещен в память)

AMD Pacific: Защищенный режим

- Запуск состоит из следующих шагов:
  - Инициализация процессора (как и при обычной загрузке за исключением некоторых флагов)
  - Защита области памяти, содержащей SLB, от доступа любой VM с помощью механизма *DEV*
  - Чтение кода *SL* и передача его в TPM для проверки (используется *специальная логика в контроллере памяти и I/O hub'е*, что делает невозможными подмены проверки программными средствами)
  - Вычисление значения хэш-функции от кода *SL* в TPM и выдача результата его сравнения с эталоном в процессор (результат помещается в специальный бит процессора)
  - Очистка Global Interrupt Flag
  - Увеличение значения указателя на вершину стека на размер SLB (также и для указателя на новый блок SLB)



AMD Pacific: Automatic Memory Clear

- **Автоматическая очистка оперативной памяти** (**AMC**) выполняется в защищенном режиме, может быть отключена в конфигурационных регистрах
- Предотвращает утечку данных, оставшихся после предыдущей сессии
- В North Bridge хранятся скрытые для системы регистры **AMC Check Registers**, содержащие информацию о последней конфигурации оперативной памяти, обновляются после каждой инициализации памяти
- Очистка выполняется:
  - После каждого *cold reset*
  - После *warm reset*, если при этом:
    - либо включена AMC
    - либо информация в AMC Check Registers не соответствует новой конфигурации памяти
- Если очистка памяти началась, то очищается вся память

AMD: резюме

- Добавлены 5 основных инструкций: **VMRUN**, **VMCALL**, **VMSAVE, VMLOAD**, **SKINIT**
- Добавлены 3 инструкции для ускорения работы: **INVLPGA**, **STGI**, **CLGI**
- Гипервизор является системным кодом
- Добавлена работа со структурами **VMCB**, описывающими переходы между гипервизором и виртуальными машинами (хранятся в оперативной памяти), а также специальная область внутренней памяти процессора, в которой хранится состояние гипервизора
- Добавлена трансляция виртуального адреса с помощью механизма **Nested Paging**, тэги **ASID** в TLB, и возможность очистки TLB для конкретной виртуальной машины
- Добавлено управление доступом в память с помощью вектора ***DEV***
- Добавлен **защищенный режим работы** с автоматической очисткой оперативной памяти

Применение аппаратной виртуализации

- Повышает изоляцию и независимость гостевых операционных систем
- При полной поддержке гипервизором повышает производительность систем
- Поддерживает консолидацию нагрузки и миграцию
- Упрощает логику и программную реализацию монитора виртуальных машин

Спасибо за внимание!